Pergamon

Computers Math. Applic. Vol. 34, No. 11, pp. 113–119, 1997

0898-1221/97 $17.00 + 0.00

PII: S0898-1221(97)00225-3

The Hidden Algorithm of Ore's Theorem on Hamiltonian Cycles

E. M. PALMER
Department of Mathematics
Michigan State University
East Lansing, MI 48824, U.S.A.
palmer@math.msu.edu

Dedicated to Oystein Ore on the 97th anniversary of his birth.

Abstract—In 1960, Ore found a simple sufficient condition for a graph to have a Hamiltonian cycle. We expose a heuristic algorithm, hidden in Ore's proof, which can be very effective in actually finding such a cycle. This algorithm is always reasonably efficient and suggests an easy proof that almost all graphs are Hamiltonian.

Keywords—Graph algorithm, Hamiltonian, Probability.

1. INTRODUCTION

Thirty-five years ago, Ore published a one page note in the *Monthly* giving a sufficient condition for a graph to be Hamiltonian [1]. His result, along with many others of the same flavor, appears in almost every text on the subject (for example, [2,3]). The usefulness of these theorems is limited by the fact that they can only be applied to graphs with most vertices of very high degree, and consequently, they predict the existence of a Hamiltonian cycle in a random graph only if the edge probability is quite large. But there can be hidden gold in a proof. In this paper, we will take a closer and more opportunistic look at Ore's note and reveal the makings of an efficient algorithm that has applicability vastly superior to the theorem. We will show how a randomized version of it could easily have been exploited at the time to prove that almost all graphs are Hamiltonian, even if the edge probability p approaches zero as the number n of vertices goes to infinity. This was an important problem left untouched by Erdős and Rényi in their blockbuster paper [4] which laid the foundations of the theory of random graphs. A decade later, Perepelica's insertion algorithm made a beginning on this problem [5]. A nice theoretical improvement of Wright [6] used a nonconstructive method for establishing the existence of a spanning cycle in almost all graphs in 1974. Finally, the discovery of the method of path transforms by Pósa [7] led to the best algorithms for finding spanning cycles. An excellent exposition of such an algorithm devised by Angluin and Valiant is available in the book by Wilf [8]. For many details on the best possible results for spanning cycle theorems and algorithms, one should study the research monograph by Bollobás [9].

I would like to thank J. G. Gimbel, my colleague and guide to the Grizzly wilderness of North America from Maine to Alaska, for his encouragement and thorough reading of this manuscript. The support of this Arctic cowboy is greatly appreciated. My spartan colleague, B. Sagan, also fiddled around with it and made several very helpful suggestions.

We provide below a few basic definitions from graph theory. For more background on random graphs, the reader may consult the introductory book [10]. It should be sufficient for the straightforward treatment of this article.

A graph G consists of a vertex set $V(G) = \{1, 2, \ldots, n\}$ and an edge set $E(G)$ which is a subset of the collection of 2-subsets of V. The cardinality n of the vertex set is called the *order* of G, while the cardinality M of the edge set is the *size* of G. If $\{u, v\}$ is an element of the edge set of G, we write uv instead of $\{u, v\}$, and we say that u and v are *adjacent* vertices. The *neighbors* of the vertex u are all the vertices adjacent to u, and the number of these is called the *degree* of u, written $\deg u$. A *cycle* is a set of r distinct vertices $u_1, \ldots, u_r$ with r at least three, such that u_1 is adjacent to u_2, u_2 is adjacent to u_3, etc., and u_r is adjacent to u_1. If the cycle contains all n vertices of the graph, it is said to *span* the vertices and is called a *spanning* or *Hamiltonian* cycle. If an edge is deleted from a spanning cycle, one obtains a *spanning* path.

As for random graphs, our sample space consists of all labeled graphs G with n vertices. Given the edge probability $0 < p < 1$, the probability of a graph G with M edges is defined by

$$P(G) = p^M (1-p)^{N-M}, \tag{1.1}$$

where $N = \binom{n}{2}$, the number of slots available for edges. Thus, the sample space consists of Bernoulli trials and the edges are selected independently with probability p. Suppose $\mathcal{A}$ is a set of graphs of order n with some specified property Q. If the probability $P(\mathcal{A})$ approaches 1 as n goes to infinity, then we say that *almost all graphs have property Q or the random graph has property Q a.s.* (almost surely). Thus, we are going all the way with Model A of [10].

2. ORE'S THEOREM

Scores of papers and books have cited Ore's 1960 article [1], as well as its precursor by Dirac [11]. It has been obtained as a corollary of several important theorems in graph theory discovered by Bondy, Chvátal, and Pósa that can be found in any graph theory text.

THEOREM 1. *[1] Let G be a graph of order $n \geq 3$. For every pair of nonadjacent vertices u, v, we assume that*

$$\deg u + \deg v \geq n. \tag{2.1}$$

Then, G is Hamiltonian.

An easy proof by contradiction that one finds in many texts goes something like this. Fix n and suppose the theorem is false. Then there must be a counterexample. Take one with a maximum number of edges, i.e., a big bad graph. This graph satisfies the degree condition (2.1), but has no spanning cycle. But, if we add an edge uv to it (any one at all will do!), it will still satisfy (2.1) and be bigger in size. So the theorem works for this new graph, and hence, it has a spanning cycle! Now the edge uv has outlived its usefulness and we delete it to obtain a spanning path from u to v back in the bad graph.

A straightforward argument using the pigeon-hole principle can be used to obtain a contradiction. Suppose w is a neighbor of v. Its successor w' on the spanning path from u to v cannot be adjacent to u. Otherwise, it is easy to find a spanning cycle in the bad graph. So there are $\deg v$ vertices other than u that are forbidden to be neighbors of u. And there are $\deg u$ vertices other than u that must be neighbors of u. Since there are only $n-1$ vertices other than u in G, there is simply not enough room for the neighbors of u and those forbidden to u, i.e., we contradict condition (2.1).

A simple procedure to examine a graph to see if it satisfies the hypothesis of Ore's theorem requires $\mathcal{O}(n^2)$ operations. We assume that the degrees of all the vertices are computed from an adjacency matrix at a cost of about n^2 operations. Then, the inspection of the degree sum of all pairs of vertices takes about the same effort. A random graph with edge probability p will

almost surely fail to satisfy the hypothesis unless $p > 1/2$ (see [12], also see [10, p. 84 and exercise 5.4.4]).

Let us contrast the proof above with Ore's approach which we now sketch. Ore suggested that we arrange the vertices in circular order with as many edges as possible on the boundary. Now suppose that there is a gap, i.e., no edge joining consecutive vertices u and v on the boundary. Suppose that u precedes v in the counterclockwise order. Then, there are two possibilities. The first is that there must be another pair of consecutive vertices w and its successor w' such that u is adjacent to w and v is adjacent to w'. It is easy to see how the chords uw and vw' of our circular arrangement can be exchanged with the gap uv and the possible gap ww' to obtain at least an additional edge on the boundary. This contradicts the assumption that the vertices were arranged with as many edges as possible on the boundary. The second possibility is that no such crossing chords exist. But this can be shown to cause a contradiction to the degree condition (2.1). Hence, there must be no gaps on the boundary at all. End of proof sketch.

These are not exactly Ore's words, but it is more or less how his note reads. Is not this approach just begging to be transmogrified into a heuristic algorithm?

3. CRISS-CROSS ALGORITHM

Here is a sketch of such an algorithm.
Given a graph,

Step 0. Arrange the vertices in a circle.

Step 1. Look around the boundary, say in the counterclockwise direction, for consecutive nonadjacent vertices, i.e., a gap. If there are no gaps, quit with the spanning cycle on the boundary. Otherwise, look for a pair of crossing chords from the vertices of the gap to some other pair of consecutive vertices that may or may not be adjacent (possible gap 2).

If found, (i.e., gap 1 was good!), simply rearrange the circular order of the vertices in the obvious way so that the two chords become edges on the boundary and the gaps are switched to the interior. Each time we play this game of criss-cross successfully, one or two gaps on the boundary of the circular arrangement of vertices are replaced by two edges. Otherwise repeat Step 1 with the next gap.

Continue until the spanning cycle is on the boundary, or until every gap is bad.

Note that we did not even bother to put a lot of edges on the boundary to start the algorithm. The complexity of the algorithm is $\mathcal{O}(n^3)$.

My undergraduate honor student and computer expert, S. Mathison, programmed the criss-cross algorithm in living color on his CompuAdd 320 and we tried it out on a few thousand cases. Here is how we did it. First, we used the following formula to calculate the edge probability p:

$$p^2 n = c \log n, \tag{3.1}$$

with constant $c > 0$. The formula will be explained later! We generated 100 graphs at random for each of the values of p determined by n and c in Table 1 and tried the criss-cross algorithm on each one. The table entry in the column under CC is the number of times the criss-cross algorithm succeeded in finding a spanning cycle.

That column of the table took about 20 minutes of computer time for each value of c, so the whole thing took about an hour. Note the dramatic increase in the success rate for fixed n with the slightly higher values of p that come from increasing c. And note how effective the algorithm is for $c = 1.5$, even for values of n as small as 20. We hasten to point out, however, that if one of the best Hamiltonian algorithms now known had been turned loose on these cases, the results might be boring, because each entry in such a table would be pretty close to 100 except for some of the smaller values of n. But those algorithms were not discovered until 15 years after Ore's

Table 1. Successful trials of criss-cross and improved criss-cross.

$c = 0.5$					
n	p	CC	Imp	CC+	Imp+
10	.3393	10	12	4	6
20	.2737	14	12	9	7
30	.2381	9	20	4	15
40	.2147	13	27	5	19
50	.1978	19	36	11	28
60	.1847	16	30	9	23
70	.1742	15	38	10	33
80	.1655	18	32	12	26
90	.1581	20	48	8	36
100	.1517	24	41	13	30
$c = 1.0$					
n	p	CC	Imp	CC+	Imp+
10	.4799	45	57	7	19
20	.3870	65	67	13	15
30	.3367	77	78	15	16
40	.3037	78	82	11	15
50	.2797	83	88	6	11
60	.2612	80	87	12	19
70	.2464	83	92	4	13
80	.2340	87	89	9	11
90	.2236	84	95	5	16
100	.2146	86	91	8	13
$c = 1.5$					
n	p	CC	Imp	CC+	Imp+
10	.5877	81	79	7	5
20	.4740	94	93	5	4
30	.4124	95	96	2	3
40	.3719	94	96	3	5
50	.3426	97	98	2	3
60	.3199	96	99	1	4
70	.3017	99	98	2	1
80	.2866	99	100	0	1
90	.2739	98	100	0	2
100	.2628	100	100	0	0

little note appeared! We leave to the reader the fun of trying out the algorithm of Angluin and Valiant mentioned above on such graphs to see how well it outperforms the criss-cross algorithm on the values of n in the table.

Recall that Ore's proof begins with the suggestion that as many edges as possible be placed around the boundary of the circular arrangement. We could enhance the starting conditions of the algorithm by using Depth-First-Search to find a long path whose edges could be put on the boundary before running criss-cross. We tried this for the same graphs we used above and you can see what happened in the column headed Imp.

The success rate for the improved algorithm is only slightly increased. But the total time taken to calculate all the numbers under Imp was less than five minutes! We also added two more calculations to show the number of times each algorithm outperformed the other. For example, when $n = 10$ and $c = 1.0$, the improved algorithm found spanning cycles in 19 graphs for which the criss-cross algorithm failed, whereas in seven graphs, the criss-cross algorithm found a spanning cycle, while the improved algorithm failed. Hence, the number of successes of the

improved algorithm should exceed that of the criss-cross algorithm by 12. This is reflected in the difference of the entries 45 and 57 of the same row.

The reader may wish to contrast Ore's note with the proof of a slightly weaker result of Dirac [11]. Is there an obvious algorithm beneath the surface of Dirac's indirect proof?

4. PROBABILISTIC ANALYSIS

First, let us investigate formula (3.1) for the edge probability that we used above in our experiments. When we arrange the vertices in circular order according to Step 0 of the criss-cross algorithm, let us assume that we use their integer values to order them from 1 to n. Then, for each graph G in the sample space, we can define the random variable $X = X(G)$ to be the number of gaps, i.e., consecutive nonadjacent vertices, in this arrangement. Thus, the expected number of gaps is

$$E[X] = n(1-p). \tag{4.1}$$

Some of these gaps are good because there are crossing chords as described in the algorithm which can be exchanged with the gap to increase the number of edges on the boundary. But a gap is bad if crossing chords are not available in any of the $n-3$ possible locations around the boundary. If $Y = Y(G)$ is the number of bad gaps, then it is easy to see that its expectation is

$$E[Y] = n(1-p)\left(1-p^2\right)^{n-3}. \tag{4.2}$$

Using the Maclaurin series of the $\log(1-p^2)$, we see that if $p^4 n \to 0$ as $n \to \infty$, then

$$E[Y] \sim ne^{-p^2 n}. \tag{4.3}$$

This equation motivates the choice of p in (3.1) above. With p so defined and $c > 1$, $E[Y] \to 0$ as $n \to \infty$, and so for almost all graphs, every gap is good at the start of the criss-cross algorithm. As soon as one gap is repaired, however, it can no longer be said that all the chords are present as edges with probability p. An argument would have to be made that the edges inside the boundary are still nearly random throughout the repair process of the algorithm.

Therefore, we take Ore's advice and prepare the graph a little by putting a lot of edges on the boundary before we start looking for crossing chords. And we do this in the simplest possible way, using Depth-First-Search, but with no back-tracking allowed.

We require a pair of beautiful inequalities that are used repeatedly in this business.

LEMMA 4.1. *For $0 < p < 1$ and any integer $n > 1$, we have*

$$1 - pn < (1-p)^n < e^{-pn}. \tag{4.4}$$

The probability that Depth-First-Search finds a path of length at least m is given by the product

$$\prod_{i=1}^{m} \left(1 - (1-p)^{n-i}\right). \tag{4.5}$$

On using the smallest factor m times and applying both inequalities of Lemma 4.1, we see that

$$\prod_{i=1}^{m} \left(1 - (1-p)^{n-i}\right) \geq 1 - me^{-p(n-m)}. \tag{4.6}$$

Hence, we see that if we set

$$k = n - m, \tag{4.7}$$

and if we can express both k and p as functions of n so that

$$\frac{m}{\exp(pk)} \to 0, \tag{4.8}$$

as $n \to \infty$, then almost all graphs have a path of length at least m.

At this point, we suggest the manner in which we shall avoid the difficulty of the loss of randomness after each successful criss-cross. Erdős and Rényi used this idea in their big paper [4, pp. 53,54] when they investigated the giant component, although they dealt with a slightly different probability model. Suppose we are given an edge probability p. We plan to divvy it up into $k+1$ parts as follows:

$$p = p_0 + kp_1, \tag{4.9}$$

and feed edges to the algorithm in small doses. It may be helpful to think of these batches of edges as having different colors. We will try to use p_0 to get enough red edges for a long path and then use p_1 to generate some blue edges for a successful criss-cross. Then, we introduce some green edges with probability p_1 to try for another successful criss-cross, continuing as many as k times if necessary. It is alright to do this because when we put in the edges separately with probabilities $p_0, p_1, p_1, \ldots, p_1$, the probability that a pair of vertices is adjacent with an edge of some color is

$$1 - (1-p_0)(1-p_1)^k, \tag{4.10}$$

which is less than p. This follows quickly from the left inequality in Lemma 4.1. So the probability of a spanning cycle of colored edges is less than the probability of a spanning cycle in Model A. Let us call this approach the improved criss-cross algorithm with randomization.

Suppose we are trying to repair a gap and so we introduce new edges with probability p_1. The probability that the algorithm fails to find crossing chords is at most

$$\left(1-p_1^2\right)^{n-3}.$$

Hence, the probability that the algorithm successfully fixes k bad gaps is at least

$$\left(1-\left(1-p_1^2\right)^{n-3}\right)^k.$$

Once again, we apply both inequalities in the lemma to find that if we can express k and p_1 as functions of n so that

$$\frac{k}{\exp\left(p_1^2 n\right)} \to 0, \tag{4.11}$$

then k gaps can be repaired in almost all graphs.

Now our task is to solve the system of conditions (4.8) with p replaced by p_0, (4.9), and (4.11) simultaneously. If p is given as in the theorem below, we take

$$p_0 = \frac{c_0 \log n}{n^{1/4}}, \tag{4.12}$$

with $c > c_0 > 1$. And for k, we take the nearest integer to $n^{1/4}$. Now it is a routine matter to check all the required conditions, and the consequences are summarized in the following theorem.

THEOREM 2. *If the edge probability is defined by*

$$p = \frac{c \log n}{n^{1/4}}, \tag{4.13}$$

with constant $c > 1$, then almost all graphs are Hamiltonian and the improved criss-cross algorithm, with randomization, almost surely finds a spanning cycle.

Most of the execution time for the improved algorithm is spent on the Depth-First-Search which requires $\mathcal{O}(n^2)$ steps. After that, there are almost surely only about $n^{1/4}$ criss-cross operations, which take not more than about n steps each. So the improved algorithm almost surely finds a spanning cycle or quits in not more than $\mathcal{O}(n^2)$ steps. We emphasize that the analysis above only goes to show that the improved algorithm almost surely finds a spanning cycle when p is defined by (4.13) and the edges it causes are released for use by the algorithm in stages.

Table 1 shows that (4.13) could probably be replaced by (3.1). But it can be shown that the value of p in the theorem cannot be substantially improved, i.e., lowered, with this method of proof. It also remains to investigate the performance of the algorithm when $c < 1$. The data indicate that even the unadulterated criss-cross algorithm works almost surely for the value of p in (3.1) with $c > 1$, while failure is quite likely if $c < 1$. But we have no proof of this either. Equation (4.1) says that there are about n gaps at the beginning of the algorithm. That is too many gaps for the method of proof to work. We would need condition (4.11) with k replaced by n, and that is impossible. So there is just not enough edge probability to divvy.

It should also be mentioned that the algorithm of Perepelica [5] is also quite straightforward and finds a spanning cycle almost surely for p defined as in (3.1) with $c > 2.0$. But no proof is provided and so this makes a nice exercise for the reader (and the writer!).

We conclude with the reminder that there are much better algorithms now known for finding Hamiltonian cycles in random graphs, and the theorem above is far from the strongest result. But the analysis for such improvements is much more difficult. For example, to establish the strongest form of the theorem that almost all graphs are Hamiltonian requires about five ingredients,

(1) a much more sophisticated long path algorithm,
(2) the threshold for connectedness,
(3) an isoperimetric inequality that guarantees that small sets of vertices have large sets of neighbors,
(4) Pósa transforms, and
(5) an approach such as that used above to regain randomness after rearranging the graph.

There are also substantial technical details to keep all the parts in harmony. Of course the payoff is big. It can be shown that with p defined by equation (3.1) with $c > 1$, but without the square (!!), almost all graphs are Hamiltonian. But even Bollobás [9] has called the method of treatment a "tortuous road". On the other hand, we have seen an easy partial solution to this hard problem of Erdős and Rényi that could have been uncovered right around the time that they founded the theory of random graphs.

REFERENCES

1. O. Ore, Note on Hamilton circuits, *Amer. Math. Monthly* **67**, 55 (1960).
2. G. Chartrand and L. Lesniak, *Graphs & Digraphs*, Wadsworth, Pacific Grove, CA, (1986).
3. F. Buckley and F. Harary, *Distance in Graphs*, Addison-Wesley, Redwood City, CA, (1990).
4. P. Erdős and A. Rényi, On the evolution of random graphs, *Magyar Tud. Acad. Mat. Kutató Int. Közl.* **5**, 17–60 (1960).
5. V.A. Perepelica, On two problems from the theory of graphs, *Soviet Math. Dokl.* **11**, 1376–1379 (1970).
6. E.M. Wright, For how many edges is a graph almost certainly Hamiltonian?, *J. London Math. Soc.* **8** (2), 44–48 (1974).
7. L. Pósa, Hamiltonian circuits in random graphs, *Discrete Math.* **1**, 359–364 (1976).
8. H.S. Wilf, *Algorithms and Complexity*, Prentice-Hall, Englewood Cliffs, NJ, (1986).
9. B. Bollobás, *Random Graphs*, Academic, London, (1985).
10. E.M. Palmer, *Graphical Evolution: An Introduction to the Theory of Random Graphs,* Wiley Inter-Science Series in Discrete Mathematics, John Wiley & Sons, New York, (1985).
11. G.A. Dirac, Some theorems on abstract graphs, *Proc. London Math. Soc.* **2** (Ser. 3), 69–81 (1952).
12. J.G. Gimbel, D. Kurtz, L. Lesniak, E.R. Scheinerman and J.C. Wierman, Hamiltonian closure in random graphs, *Annals of Discrete Math.* **33**, 59–67 (1987).